

Graphics Debugging and Profiling

Using Nsight

Overview

- Download and install Nsight
- Debugging
 - Toolbar and features
 - API Inspector
 - Nsight Window
- Profiling
 - Start application trace
 - Generate Summary report
 - Report Breakdown

Download app

[Home](#) > [GameWorks](#) > [Tools](#) > [Nsight Visual Studio Edition 5.3 New Features](#)

Nsight Visual Studio Edition 5.3 New Features

NVIDIA® Nsight™ Visual Studio Edition 5.3 is available for download under the NVIDIA Registered Developer Program. This release provides developers frame debugging and profiling support for Direct3D 9/11/12, OpenGL, and Vulkan applications with newly added support for OpenVR and HTC Vive, Visual Studio 2017, Microsoft Hybrid laptops, and the latest Pascal GPUs.

This release supports [CUDA Toolkit 8.0](#).

[Register](#) for free access to latest [Nsight™ Visual Studio Edition](#) releases.

For a complete overview of all Nsight™ Visual Studio Edition features and access to resources, please visit the main [Nsight™ Visual Studio Edition](#) page.

[Download >](#)[Documentation >](#)

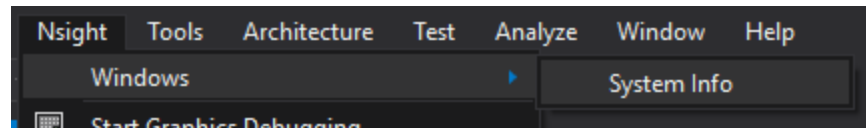
- Download it from here
 - https://developer.nvidia.com/nsight-visual-studio-edition-5_3-new-features

Installation






- Install the application from the download location.
- Once installed there will be a Nsight tab on visual studio

System Info

- Goto Nsight -> Windows -> System Info to get cpu and gpu capabilities.

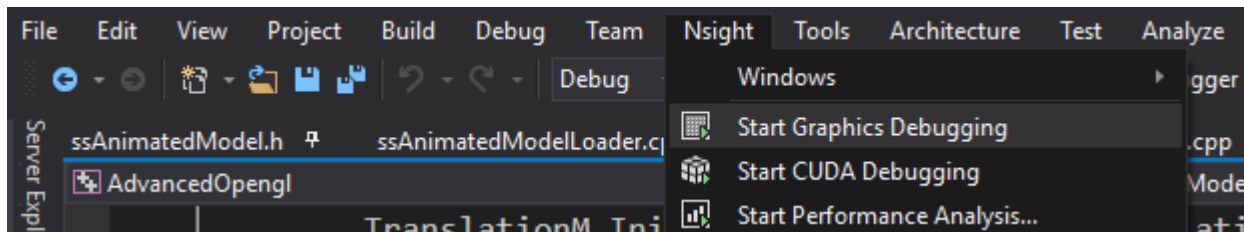


Connection Name: DESKTOP-V023A4G

 System	Attribute	Value
 Display Devices	^ CPU	
 GPU Devices	Name	Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz
 CUDA Devices	Architecture	x64
 OpenCL Devices	Frequency	3,500 MHz
	Number of Cores	4
	Page Size	4,096
	Total Physical Memory	16,225.00 MB
	Available Physical Memory	8,926.00 MB

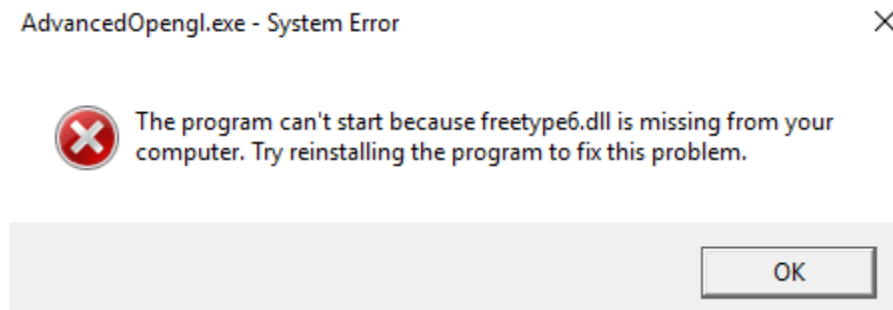
Running the App

- After building your project, click on it and press Start Graphics Debugging.



Dll missing error

- You might get an error saying the dlls are missing.



- Add all the required dlls in the release or debug folder.
- Run the graphics debugging again.

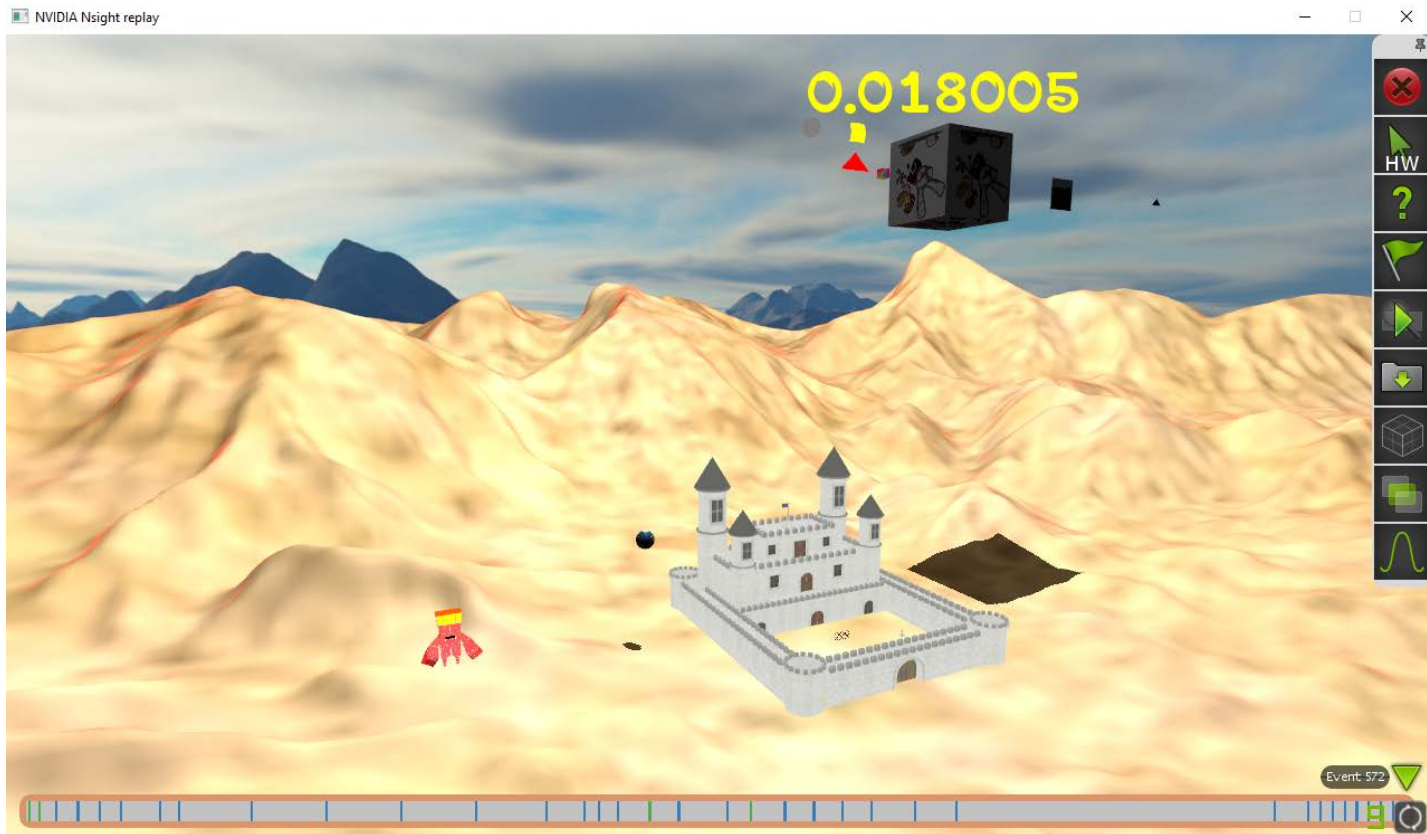
Debugging

- Once application has started. Move the camera around to see the specific object that you want to debug.
- Press Ctrl + z to to pause the current scene.
- 4 tabs will be visible on screen.






Debugging

- Press the Pause and Capture Frame button
- Or press the spacebar.



Toolbar

Toolbar

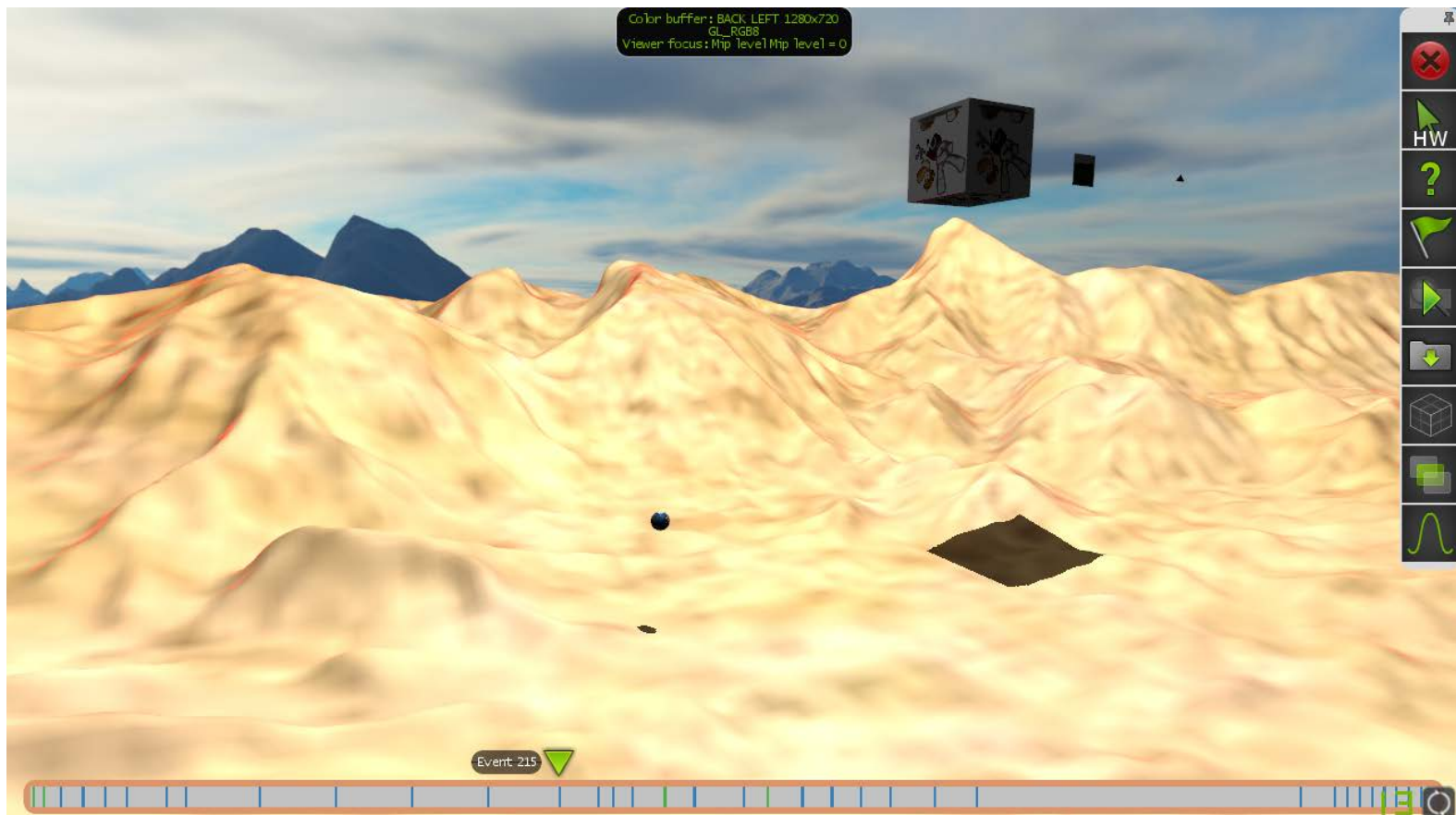
- The toolbar on the right will expand with more features.
- With options to look at different buffers  and see the current rendered object in wireframe mode  .
- If you would like to resume, press the play button on the panel on the right 

Toolbar

- The timeline below shows all objects being rendered per frame.
- The timeline has a green cursor which can be moved back and forth through each event.
- An event can be a clear screen or an object getting drawn.

Toolbar

- Here event 215 is the drawing of the terrain



Toolbar

- To see object in wireframe mode click on the cube and select red wireframe.
- Once done click disable wireframe



Toolbar

- To look at the depth buffer click on the 3 windows icon and select depth buffer.
- Back left shows the backbuffer.
- The stencil buffer can also be seen if in use. Otherwise it will all be black.



Toolbar

- The question mark button will show the keyboard shortcuts

A screenshot of a dark-themed window titled "---- HUD Shortcuts ----" with a close button in the top right corner. The window lists various keyboard shortcuts categorized into sections: General, Experiments, Scrubber, Texture, Render Target, and Display. Each section has a header line (e.g., "---General---") followed by a list of shortcuts and their functions. The shortcuts include V (Hide/show HUD components), H (Hide/show help), C (Toggle cursors), F4 (Screenshot), S (Serialization), Space (Leave debugger), Ctrl+T (2x2 textures), Ctrl+W (Wireframe), Ctrl+X (Null scissor rect), Ctrl+M (Minimum geometry), Ctrl+O (Shader overrides), Ctrl+A (Toggle all experiments), E (Profile experiment), Ctrl+Left Arrow (Previous action), Ctrl+Right Arrow (Next action), Home (First action), End (Last action), F2 (Show less info), F3 (Show more info), Ctrl+Plus (Increase zoom), Ctrl+Minus (Decrease zoom), Ctrl+0 (No zoom), Shift+Ctrl (Pixel inspection), W (Wireframe mode), N (Render targets), O (Normalization),] (Increase text size), [(Decrease text size), and Shift+F3 (Perfmarker ranges).

```
---- HUD Shortcuts ----
---General---
V      - Hide or show the HUD components
H      - Hide or show help for HUD keyboard bindings
C      - Toggle software/hardware cursors
F4     - Take screenshot of HUD
S      - Attempt serialization on captured frame
Space  - Leave frame debugger mode

---Experiments---
Ctrl+T  - 2x2 textures
Ctrl+W  - Wireframe
Ctrl+X  - Null scissor rect
Ctrl+M  - Minimum geometry
Ctrl+O  - Shader overrides
Ctrl+A  - Toggle all experiments
E       - Attempt profile experiment on captured frame

---Scrubber---
Ctrl+Left Arrow - Go to previous action
Ctrl+Right Arrow - Go to next action
Home       - Go to first action
End        - Go to last action
F2         - Current event: show less info
F3         - Current event: show more info

---Texture---
Ctrl+Plus  - Increase zoom
Ctrl+Minus - Decrease zoom
Ctrl+0     - Apply no zoom (1:1 pixel mapping)
Shift+Ctrl - Activate pixel inspection

---Render Target---
W         - Cycle last-draw wireframe mode
N         - Cycle bound render targets and depth-stencil targets
O         - Switch to the next normalization option for this texture

---Display---
]         - Increase text size
[         - Decrease text size
Shift+F3  - Show or hide perfmarker ranges
```


API Inspector

API Inspector

- With the debugger still running go to visual studio.
- Click on the API Inspector tab.
- This will show the full pipeline for the current object in the current event.
- There will be a timeline at the top for the events in the current frame and the pipeline break down at the bottom.

API Inspector

Scrubber

Scaling: Event ID

Event ID: 0 20 40 60 80 100 120 140 160 180 200 215 240 260 280 300 320 340 360 380

All Events

Render Target ... 158

Add...

API Inspector

ssAnimatedModel.h ssAnimatedModelLoader.cpp ssAnimatedModel.cpp Source.cpp

Vertex Specification

Vtx Spec

Transform

VS

TCS

TES

GS

XFB

Raster

FS

Pix Ops

FB

CS

Textures

Images

Buffers

Program

Pixels

Misc

glDrawElements(Glenum mode = GL_TRIANGLES, GLsizei count = 1572864, Glenum type = GL_UNSIGNED_INT, GLvoid* indices = 0x00000000)

Vertex Array Object

Name: 20

GL_ELEMENT_ARRAY_BUFFER_BINDING: 37

GL_VERTEX_ATTRIB_ARRAY_UNIFIED_NV: GL_FALSE

GL_ELEMENT_ARRAY_UNIFIED_NV: GL_FALSE

Vertex Attributes							
Index	Enabled	Binding	Size	Type	Relative Offset	Properties	
0	GL_TRUE	0	3	GL_FLOAT	0	Not normalized	
1	GL_TRUE	1	2	GL_FLOAT	0	Not normalized	
2	GL_TRUE	2	3	GL_FLOAT	0	Not normalized	

Vertex Bindings					
Index	Buffer	Pointer	Stride	Divisor	
0	36	0	32	0	
1	36	12	32	0	
2	36	20	32	0	

Vertex Array Data

GL_ARRAY_BUFFER_BINDING: 0

GL_DRAW_INDIRECT_BUFFER_BINDING: 0

GL_PRIMITIVE_RESTART: GL_FALSE

GL_PRIMITIVE_RESTART_INDEX: 0

Generic Attributes

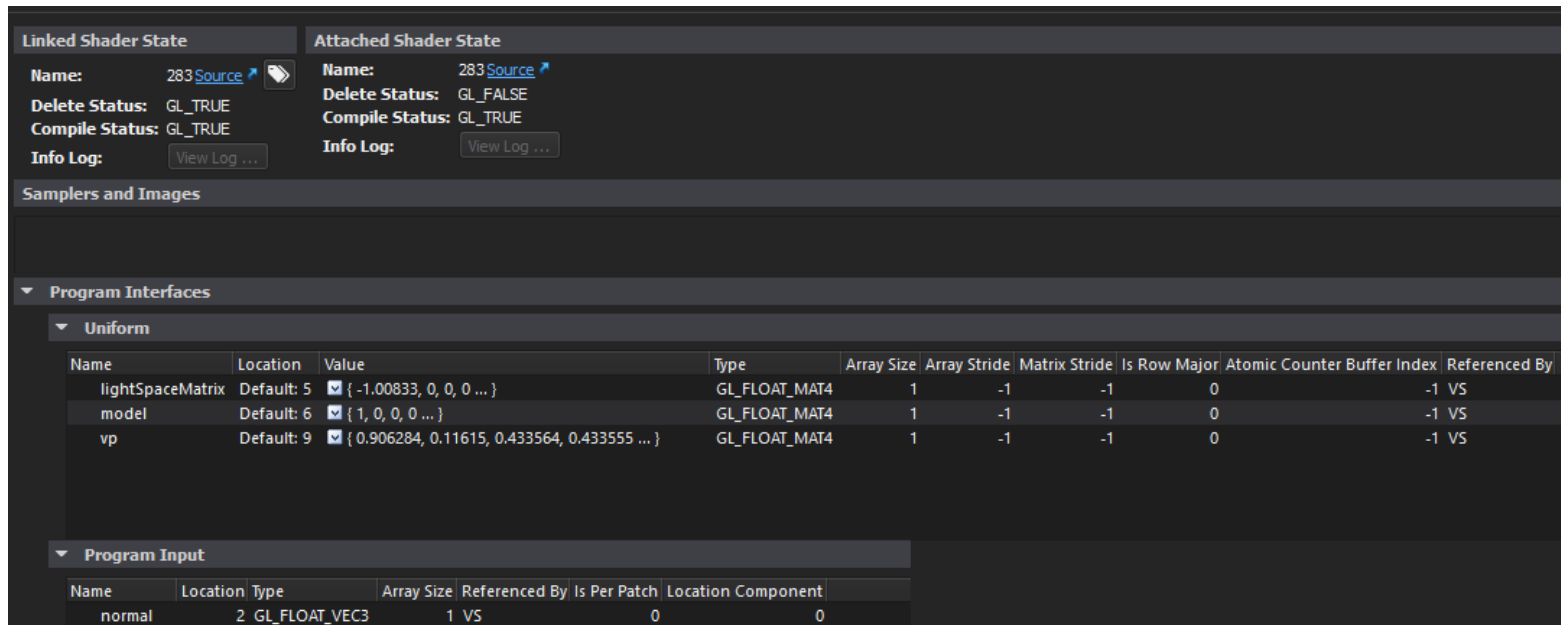
Index	Type	Normalized	X	Y	Z	W
[0 - 15]	GL_FLOAT	GL_FALSE	0.00	0.00	0.00	1.00

API Inspector

- Vtx Specs
 - Shows the vertex specifications for the current object.
 - Number of elements drawn and their type.
 - Vertex Attributes used. If they are enabled or not and to which index are they bound to.
- Transform
 - If Depth clamp enabled
 - Origin for the scene (bottom left by default)

API Inspector

- VS – Vertex Shader



The screenshot displays the API Inspector interface for a Vertex Shader (VS). The interface is organized into several sections:

- Linked Shader State:** Shows the Name (283 Source), Delete Status (GL_TRUE), Compile Status (GL_TRUE), and Info Log (View Log ...).
- Attached Shader State:** Shows the Name (283 Source), Delete Status (GL_FALSE), Compile Status (GL_TRUE), and Info Log (View Log ...).
- Samplers and Images:** A section for managing samplers and images.
- Program Interfaces:** A section for managing program interfaces, including a table of uniform variables.
- Program Input:** A section for managing program input, including a table of input variables.

The **Uniform** table under Program Interfaces shows the following data:

Name	Location	Value	Type	Array Size	Array Stride	Matrix Stride	Is Row Major	Atomic Counter	Buffer Index	Referenced By
lightSpaceMatrix	Default: 5	<input checked="" type="checkbox"/> { -1.00833, 0, 0, 0 ... }	GL_FLOAT_MAT4	1	-1	-1	0			-1 VS
model	Default: 6	<input checked="" type="checkbox"/> { 1, 0, 0, 0 ... }	GL_FLOAT_MAT4	1	-1	-1	0			-1 VS
vp	Default: 9	<input checked="" type="checkbox"/> { 0.906284, 0.11615, 0.433564, 0.433555 ... }	GL_FLOAT_MAT4	1	-1	-1	0			-1 VS

The **Program Input** table shows the following data:

Name	Location	Type	Array Size	Referenced By	Is Per Patch	Location Component
normal	2	GL_FLOAT_VEC3	1	VS	0	0

- Click on source to open source file
- Show uniform information. Location, value, type etc.

API Inspector

- TCS, TES – Are shown if tessellation shaders are used.
- GS – Is shown when Geometry shaders are used.
- XFB – Transform feedback if used.
- Raster
 - Point state and Line state. Shows size and width of point and line
 - Cull State. Define CullFaceMode (GL_BACK) and Front Face (CCW).
 - Polygon state (Smooth, Mode)
 - Multisampling State (Enabled)

API Inspector

- FS – Fragment Shader

The screenshot displays the API Inspector interface for a Fragment Shader (FS). The interface is divided into several sections:

- Linked Shader State:** Shows the Name (284 Source), Delete Status (GL_TRUE), Compile Status (GL_TRUE), and Info Log (View Log ...).
- Attached Shader State:** Shows the Name (284 Source), Delete Status (GL_FALSE), Compile Status (GL_TRUE), and Info Log (View Log ...).
- Samplers and Images:** Displays two textures: "Texture" (Unit: 0) and "shadowMap" (Unit: 1).
- Program Interfaces:** A table listing uniform variables and their values.
- Program Output:** A table showing the output of the fragment shader.

Uniform Variables Table:

Name	Location	Value	Type	Array Size	Array Stride	Matrix Stride	Is Row Major
Texture	Default: 0	0	GL_SAMPLER_2D	1	-1	-1	0
ambientStrength	Default: 1	{ 0.5 }	GL_FLOAT	1	-1	-1	0
cameraPos	Default: 2	{ -45.6451, 42.1754, 70.1176 }	GL_FLOAT_VEC3	1	-1	-1	0
lightColor	Default: 3	{ 1, 1, 1 }	GL_FLOAT_VEC3	1	-1	-1	0
lightPos	Default: 4	{ 0, 150, 0 }	GL_FLOAT_VEC3	1	-1	-1	0
shadowMap	Default: 7	1	GL_SAMPLER_2D	1	-1	-1	0
specularStrength	Default: 8	{ 0.1 }	GL_FLOAT	1	-1	-1	0

Program Output Table:

Name	Location	Type	Array Size	Referenced By	Location Index	Is Per Patch	Location Component
fragColor	0	GL_FLOAT_VEC4	1	FS	0	0	0

API Inspector

- Pix Op
 - Shows various tests like Depth, Stencil and scissor and their states, function and values .

The screenshot displays the API Inspector interface with the following sections:

- Depth State**
 - GL_DEPTH_TEST: GL_TRUE
 - GL_DEPTH_FUNC: GL_LESS
 - GL_DEPTH_WRITE_MASK: GL_TRUE
 - GL_DEPTH_CLEAR_VALUE: 1.00
 - GL_DEPTH_BOUNDS_TEST_EXT: GL_FALSE
 - GL_DEPTH_BOUNDS_EXT (zmin): 0.0000
 - GL_DEPTH_BOUNDS_EXT (zmax): 1.0000
- Stencil State**
 - GL_STENCIL_TEST: GL_FALSE
 - GL_STENCIL_FUNC: GL_EQUAL
 - GL_STENCIL_VALUE_MASK: 0x000000ff
 - GL_STENCIL_REF: 0x00000001
 - GL_STENCIL_BACK_FUNC: GL_EQUAL
 - GL_STENCIL_BACK_VALUE_MASK: 0x000000ff
 - GL_STENCIL_BACK_REF: 0x00000001
 - GL_STENCIL_WRITE_MASK: 0x000000ff
 - GL_STENCIL_BACK_WRITE_MASK: 0x000000ff
 - GL_STENCIL_CLEAR_VALUE: 0x00000000
- Stencil Ops**
 - GL_STENCIL_FAIL: GL_KEEP
 - GL_STENCIL_PASS_DEPTH_FAIL: GL_KEEP
 - GL_STENCIL_PASS_DEPTH_PASS: GL_REPLACE
 - GL_STENCIL_BACK_FAIL: GL_KEEP
 - GL_STENCIL_BACK_PASS_DEPTH_FAIL: GL_KEEP
 - GL_STENCIL_BACK_PASS_DEPTH_PASS: GL_REPLACE
- Draw Buffer Settings**

Index	Write Mask	Blend	Source	Destination	Equation
[0 - 7]	[R] GL_TRUE [G] GL_TRUE [B] GL_TRUE [A] GL_TRUE	GL_FALSE	[RGB] GL_SRC_ALPHA [A] GL_SRC_ALPHA	[RGB] GL_ONE_MINUS_SRC_ALPHA [A] GL_ONE_MINUS_SRC_ALPHA	[RGB] GL_FUNC_ADD [A] GL_FUNC_ADD
- Scissor Settings**

Index	Enabled	Scissor Box
[0 - 15]	GL_FALSE	[x] 0 [y] 0 [width] 1280 [height] 720
- Blending State**
 - GL_BLEND_ADVANCED_COHERENT_NV: GL_TRUE
 - GL_BLEND_PREMULTIPLIED_SRC_NV: GL_TRUE
 - GL_BLEND_OVERLAP_NV: GL_UNCORRELATED_NV
 - GL_BLEND_COLOR: [Color Picker]
 - GL_COLOR_CLEAR_VALUE: [Color Picker]
- Misc State**
 - GL_FRAMEBUFFER_SRGB: GL_FALSE
 - GL_DITHER: GL_TRUE
 - GL_COLOR_LOGIC_OP: GL_FALSE
 - GL_LOGIC_OP_MODE: GL_COPY

API Inspector

- FB – Frame Buffer formats and dimensions for each buffer color, depth and stencil.

The screenshot shows the 'Framebuffer Settings' panel in the API Inspector. It is divided into three main sections: 'Draw Framebuffer', 'Read Framebuffer', and 'Framebuffer Settings'.

Draw Framebuffer

Name: 0 (Default)

Drawable	DrawBuffer	Dimensions	Color Format	Depth Format
GL_FRONT_LEFT Color Depth Stencil	-	1280 x 720	<input checked="" type="checkbox"/> GL_RGB8	<input checked="" type="checkbox"/> GL_DEPTH24_STENCIL8
GL_BACK_LEFT Color Depth Stencil	+ (Index=0)	1280 x 720	<input checked="" type="checkbox"/> GL_RGB8	<input checked="" type="checkbox"/> GL_DEPTH24_STENCIL8

Read Framebuffer

Name: 0 (Default)

Drawable	ReadBuffer	Dimensions	Color Format	Depth Format
GL_FRONT_LEFT Color Depth Stencil	-	1280 x 720	<input checked="" type="checkbox"/> GL_RGB8	<input checked="" type="checkbox"/> GL_DEPTH24_STENCIL8
GL_BACK_LEFT Color Depth Stencil	+	1280 x 720	<input checked="" type="checkbox"/> GL_RGB8	<input checked="" type="checkbox"/> GL_DEPTH24_STENCIL8

Framebuffer Settings

Double Buffer: GL_TRUE
Stereo: GL_FALSE
SampleBuffers: 1

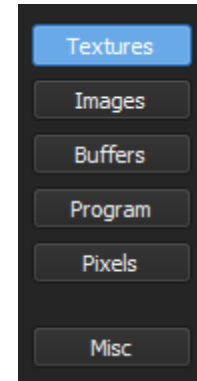
Color Read Type: GL_UNSIGNED_BYTE
Color Read Format: GL_RGB

Sample Positions

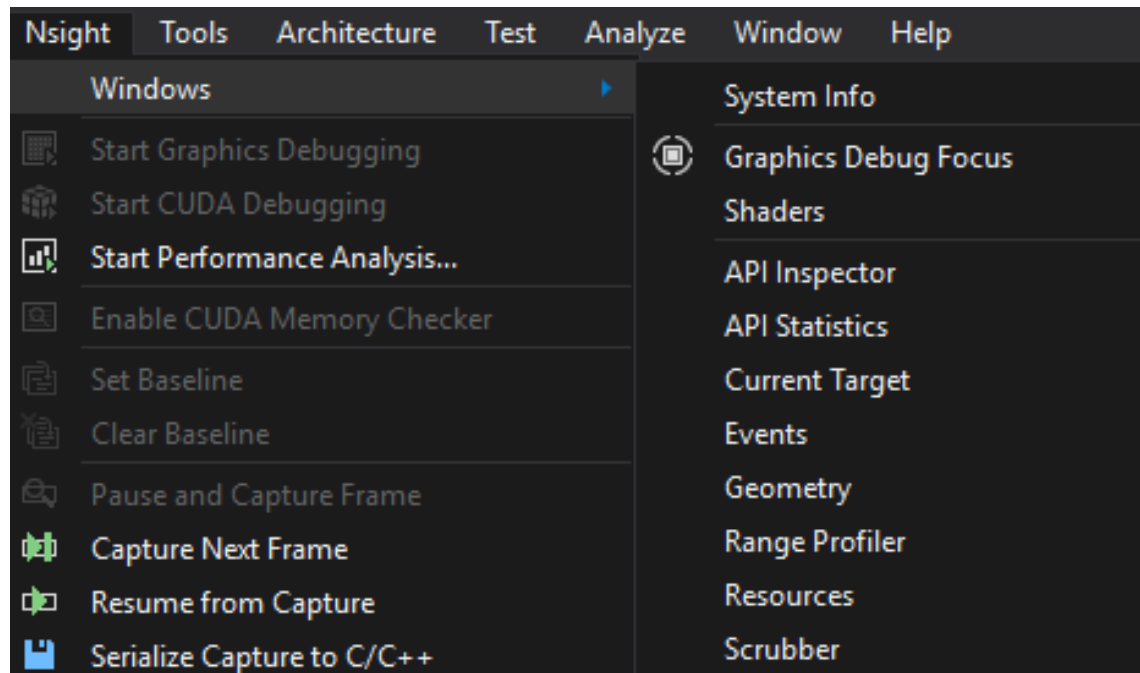
X	Y
0.38	0.88
0.88	0.63
0.13	0.38
0.63	0.13

API Inspector

- CS – Is shown if compute shaders are used.
- Textures, Images, Buffers, Program, Pixel and Misc. Give a summary of the selected item.

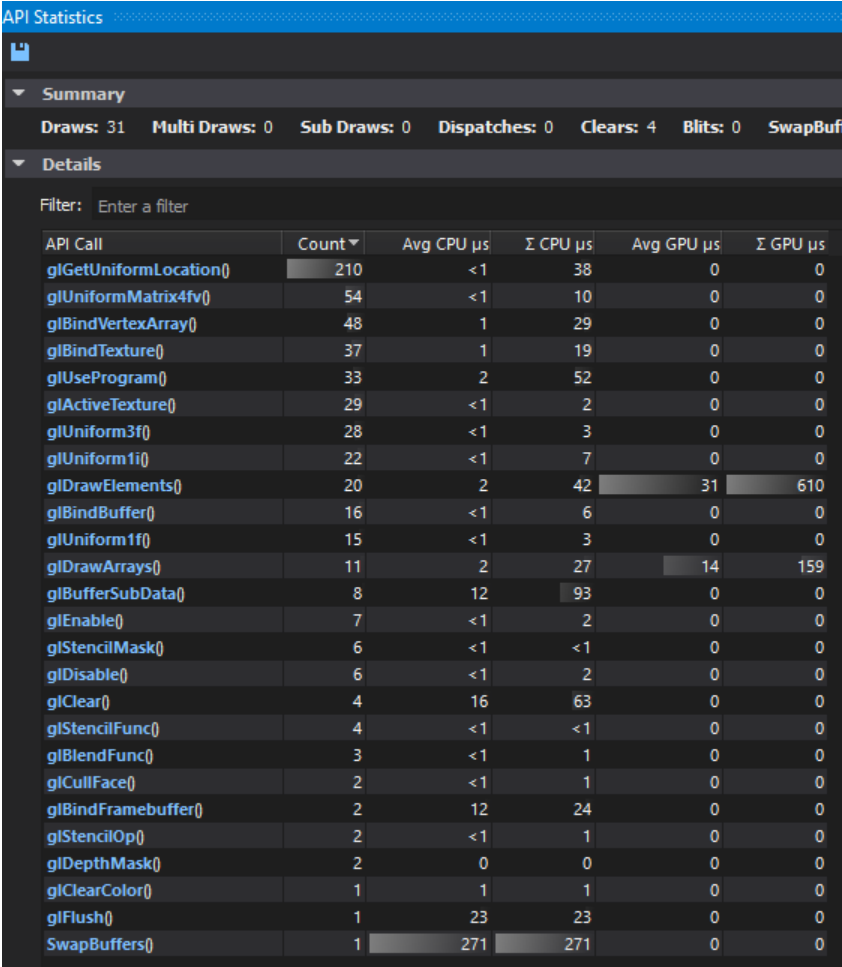


Nsight Windows



Nsight Windows

- API Statistics – Shows the details of each of the API call being made



API Statistics

Summary

Draws: 31 Multi Draws: 0 Sub Draws: 0 Dispatches: 0 Clears: 4 Blits: 0 Swap Buffers

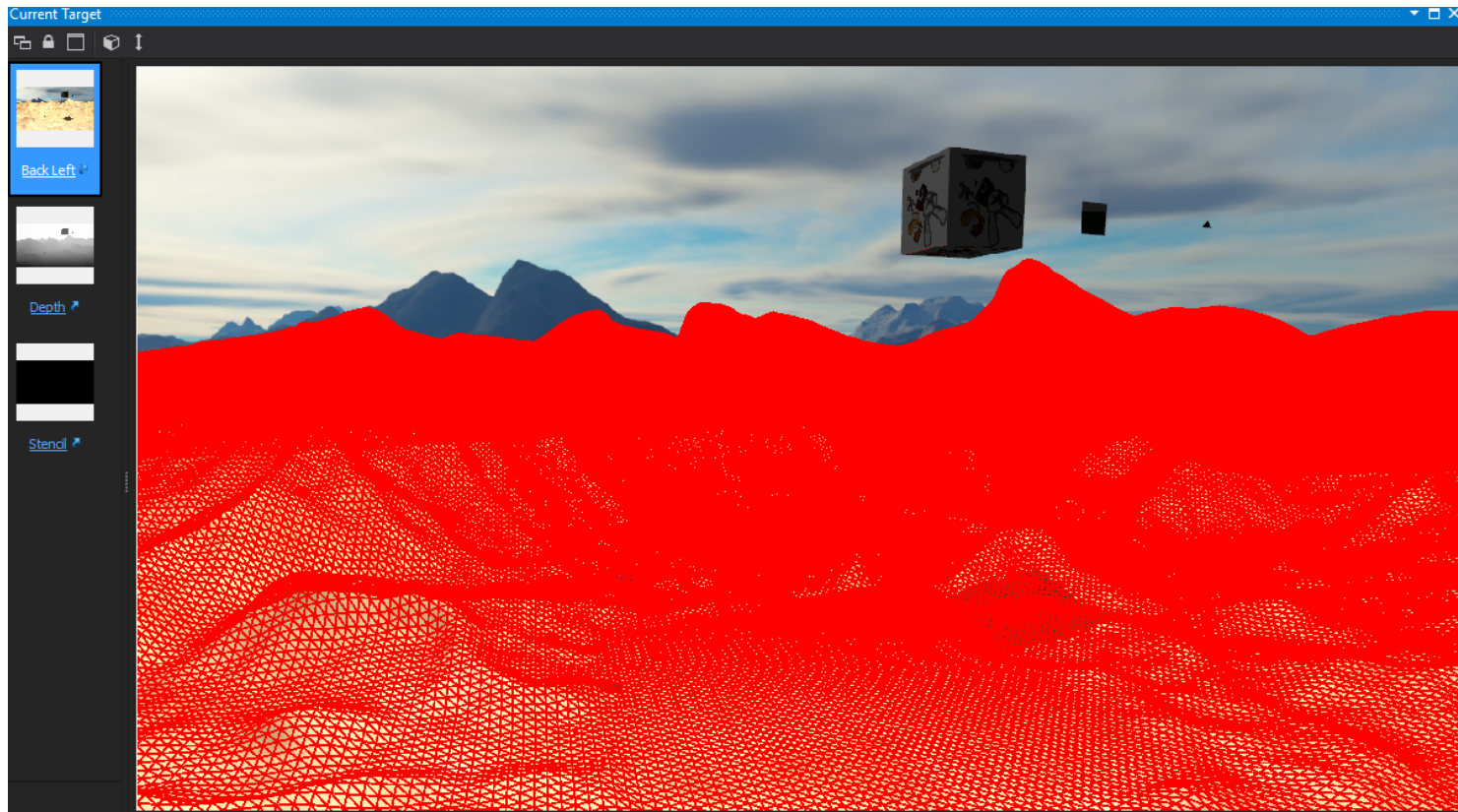
Details

Filter: Enter a filter

API Call	Count	Avg CPU μ s	Σ CPU μ s	Avg GPU μ s	Σ GPU μ s
glGetUniformLocation()	210	<1	38	0	0
glUniformMatrix4fv()	54	<1	10	0	0
glBindVertexArray()	48	1	29	0	0
glBindTexture()	37	1	19	0	0
glUseProgram()	33	2	52	0	0
glActiveTexture()	29	<1	2	0	0
glUniform3f()	28	<1	3	0	0
glUniform1i()	22	<1	7	0	0
glDrawElements()	20	2	42	31	610
glBindBuffer()	16	<1	6	0	0
glUniform1f()	15	<1	3	0	0
glDrawArrays()	11	2	27	14	159
glBufferSubData()	8	12	93	0	0
glEnable()	7	<1	2	0	0
glStencilMask()	6	<1	<1	0	0
glDisable()	6	<1	2	0	0
glClear()	4	16	63	0	0
glStencilFunc()	4	<1	<1	0	0
glBlendFunc()	3	<1	1	0	0
glCullFace()	2	<1	1	0	0
glBindFramebuffer()	2	12	24	0	0
glStencilOp()	2	<1	1	0	0
glDepthMask()	2	0	0	0	0
glClearColor()	1	1	1	0	0
glFlush()	1	23	23	0	0
SwapBuffers()	1	271	271	0	0

Nsight Windows

- Current Target – Shows Color, Depth and Stencil buffers of the current event



Nsight Windows

- Events – Shows all the events in the scene

Events

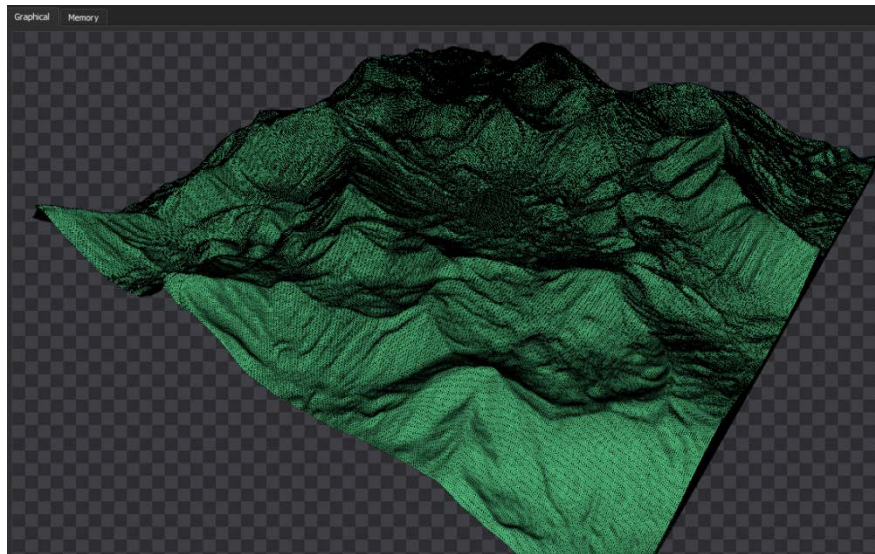
View: Hierarchical Arguments: Variable + Value Marker API:

Event: 215 Filter: Enter a filter or select a predefined one on the right

	Ever	Issues (45)	Description	Object	CPU μs	GPU μs	Thread
	0		// Start of Capture	0x000200...	-	-	4440
	1		glClear(GLbitfield mask = GL_COLOR_BUFFER_BIT GL_DEPTH_BUFFER_BIT ...	0x000200...	40	0	4440
	2		glClearColor(GLclampf red = 0.500000, GLclampf green = 0.500000, GLclam...	0x000200...	1	-	4440
	3		glCullFace(GLenum mode = GL_FRONT)	0x000200...	1	-	4440
	4		glBindFramebuffer(GLenum target = GL_FRAMEBUFFER, GLuint framebuff...	0x000200...	13	-	4440
	5		glClear(GLbitfield mask = GL_DEPTH_BUFFER_BIT)	0x000200...	11	0	4440
	6		glUseProgram(GLuint program = '300')	0x000200...	4	-	4440
	7		glGetUniformLocation(GLuint program = '300', GLchar* name = 'lightSpa...	0x000200...	1	-	4440
	8		glUniformMatrix4fv(GLint location = 0, GLsizei count = 1, GLboolean tran...	0x000200...	1	-	4440
	9		glGetUniformLocation(GLuint program = '300', GLchar* name = "model") ...	0x000200...	<1	-	4440
	10		glUniformMatrix4fv(GLint location = 1, GLsizei count = 1, GLboolean tran...	0x000200...	0	-	4440
	11		glBindVertexArray(GLuint array = '2')	0x000200...	2	-	4440
	12		glDrawElements(GLenum mode = GL_TRIANGLES, GLsizei count = 3, GLen...	0x000200...	3	1	4440
	13		glBindVertexArray(GLuint array = 0)	0x000200...	1	-	4440
	14		glUseProgram(GLuint program = 0)	0x000200...	1	-	4440
	15		glUseProgram(GLuint program = '303')	0x000200...	1	-	4440
	16		glGetUniformLocation(GLuint program = '303', GLchar* name = 'lightSpa...	0x000200...	1	-	4440
	17		glUniformMatrix4fv(GLint location = 0, GLsizei count = 1, GLboolean tran...	0x000200...	<1	-	4440
	18		glGetUniformLocation(GLuint program = '303', GLchar* name = "model") ...	0x000200...	0	-	4440
	19		glUniformMatrix4fv(GLint location = 1, GLsizei count = 1, GLboolean tran...	0x000200...	0	-	4440
	20		glBindVertexArray(GLuint array = '3')	0x000200...	1	-	4440
	21		glDrawElements(GLenum mode = GL_TRIANGLES, GLsizei count = 6, GLen...	0x000200...	2	5	4440
	22		glBindVertexArray(GLuint array = 0)	0x000200...	<1	-	4440
	23		glUseProgram(GLuint program = 0)	0x000200...	1	-	4440

Nsight Windows

- Geometry – Show the geomtry info and values of all the attributes passed into the shader in the Memory tab



IB Offset	Index	Index + Base	Index: 0 Type: GL_FLOAT Size: 3			Index: 1 Type: GL_FLOAT Size: 3			Index: 2 Type: GL_FLOAT Size: 3		
			0	1	2	0	1	2	0	1	2
0	0	0	-256.000	nan	256.000	0.000	0.000	0.000	1.000	0.000	0.000
1	1	1	-255.000	nan	256.000	0.002	0.000	0.000	1.000	0.000	0.000
2	513	513	-256.000	nan	255.000	0.000	0.002	0.000	1.000	0.000	0.000
3	513	513	-256.000	nan	255.000	0.000	0.002	0.000	1.000	0.000	0.000
4	1	1	-255.000	nan	256.000	0.002	0.000	0.000	1.000	0.000	0.000
5	514	514	-255.000	37.333	255.000	0.002	0.002	0.000	1.000	0.000	0.000
6	1	1	-255.000	nan	256.000	0.002	0.000	0.000	1.000	0.000	0.000
7	2	2	-254.000	nan	256.000	0.004	0.000	0.000	1.000	0.000	0.000
8	514	514	-255.000	37.333	255.000	0.002	0.002	0.000	1.000	0.000	0.000
9	514	514	-255.000	37.333	255.000	0.002	0.002	0.000	1.000	0.000	0.000
10	2	2	-254.000	nan	256.000	0.004	0.000	0.000	1.000	0.000	0.000
11	515	515	-254.000	37.956	255.000	0.004	0.002	0.000	1.000	0.000	0.000
12	2	2	-254.000	nan	256.000	0.004	0.000	0.000	1.000	0.000	0.000
13	3	3	-253.000	nan	256.000	0.006	0.000	0.000	1.000	0.000	0.000
14	515	515	-254.000	37.956	255.000	0.004	0.002	0.000	1.000	0.000	0.000
15	515	515	-254.000	37.956	255.000	0.004	0.002	0.000	1.000	0.000	0.000
16	3	3	-253.000	nan	256.000	0.006	0.000	0.000	1.000	0.000	0.000
17	516	516	-253.000	38.539	255.000	0.006	0.002	0.000	1.000	0.000	0.000
18	3	3	-253.000	nan	256.000	0.006	0.000	0.000	1.000	0.000	0.000
19	4	4	-252.000	nan	256.000	0.008	0.000	0.000	1.000	0.000	0.000
20	516	516	-253.000	38.539	255.000	0.006	0.002	0.000	1.000	0.000	0.000
21	516	516	-253.000	38.539	255.000	0.006	0.002	0.000	1.000	0.000	0.000
22	4	4	-252.000	nan	256.000	0.008	0.000	0.000	1.000	0.000	0.000
23	517	517	-252.000	39.161	255.000	0.008	0.002	0.000	1.000	0.000	0.000

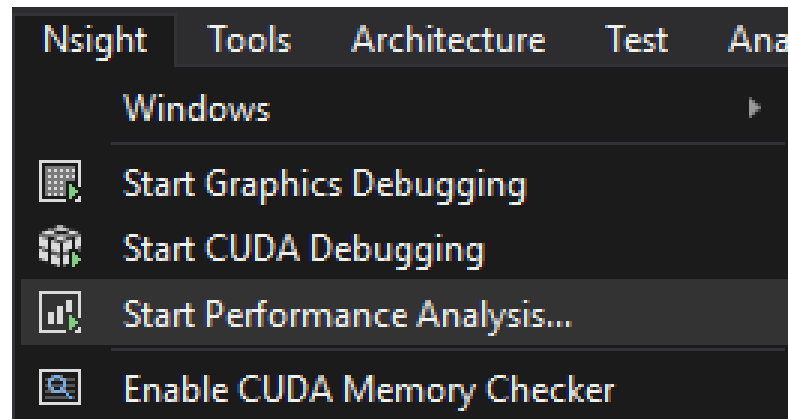
Nsight Windows

- Resources – shows all the resources used in the scene textures, shaders, buffers, etc.
- Scrubber – Shows the time line in a separate window.

Profiling

Profiling

- Press the stop button to stop the graphics debugging process.
- Goto Nsight -> Start Performance Analysis for start the profiling



Profiling

- Select OpenGL in the trace settings and press Launch in Application Control

Application SettingsConn: localhost App: AdvancedOpengl.exe Args: Sync: True

Connection Name:localhost

Application:C:\Users\siddesktop\Desktop\opengl-mds-tutorials\Advanced\Advanced_OpenGL\Debug\AdvancedOpengl.exe

Arguments:

Working Directory:C:\Users\siddesktop\Desktop\opengl-mds-tutorials\Advanced\Advanced_OpenGL\Advanced\

☒ Remote Options

Triggers and ActionsStart: On Process Launch Stop: Manually

Activity TypeTrace Application

Trace SettingsOpenGL

☐ System(2/5) CPU Thread Trace, Module Trace

☐ Tools Extension(4/4) Markers, Push/Pop Ranges, Start/End Ranges, Resource Naming

☐ CUDA(4/4) Driver API Trace, Runtime API Trace, Software Counters, Kernel Launches and Memory Operations, Host Callback Trace

☐ OpenGL(3/3) API Trace, Resource Trace, Program Source Code, Program Build Callback Trace, Program Binary Code, Reference Counter, Command Trace


☐ DirectX(9/11) API Trace, CPU Frames, GPU Frames, Push Buffers, Shader Compiles, Performance Markers, Performance Ranges, User Annotation Markers, User Annotation Ranges

☒ OpenGL(6/6) API Trace, CPU Frames, GPU Frames, Push Buffers, Debug Event Markers, Debug Event Ranges

☐ OpenVR(1/1) API Trace (as Tools Extension Start/End Ranges)


Advanced OptionsETW: Default NVE: Default

Connection Status




Available Devices:
GeForce GTX 970 (GM204)

Application Control



Launch
Kill

Capture Control

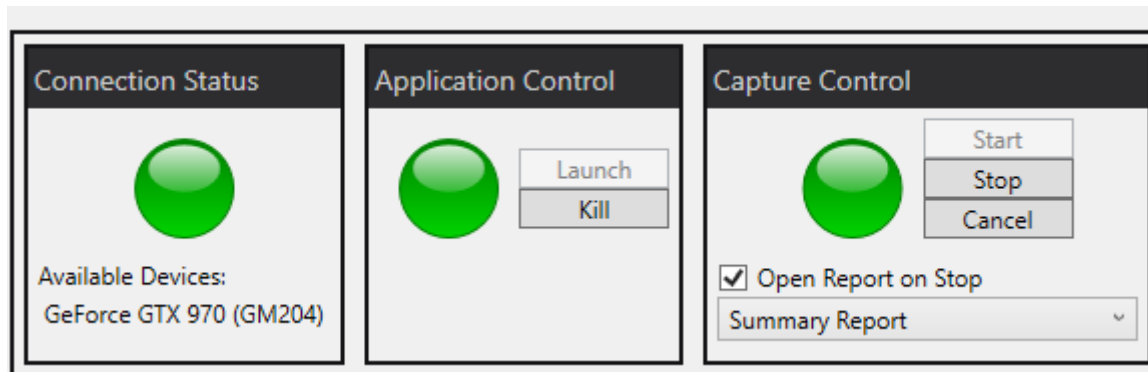


Start
Stop
Cancel

☒ Open Report on Stop
Summary Report

Profiling

- Let the application run for 5 ~10 seconds and then press the stop button in the capture control.



- Once stopped the summary report will be displayed

Profiling

- A summary report is presented

The screenshot shows the NVIDIA Nsight Visual Studio Edition interface. The top toolbar includes a 'Summary Report' button. The main content area is divided into two sections: 'Session Overview' and 'OpenGL Overview'.

Session Overview

Summary of session information related to the captured data.

- [Session Summary](#)
- [Timeline](#)
- [Activity](#)

AdvancedOpengl.exe (Trace Application)
Captured 6.83 seconds of data on 7/23/2017 10:11:35 AM

Arguments:
Working Dir: C:\Users\siddesktop\Desktop\opengl-mds-tutorials\Advanced\Advanced_OpenGL\Advanced\
Connection: localhost

OpenGL Overview

Summary of captured OpenGL activity.

- [API Calls | Summary](#)
- [Draw Calls | Transfers](#)
- [Frames | Dispatches](#)

CPU All Cores

API Time (%)	Min	Avg	Max
31.3		656.5	

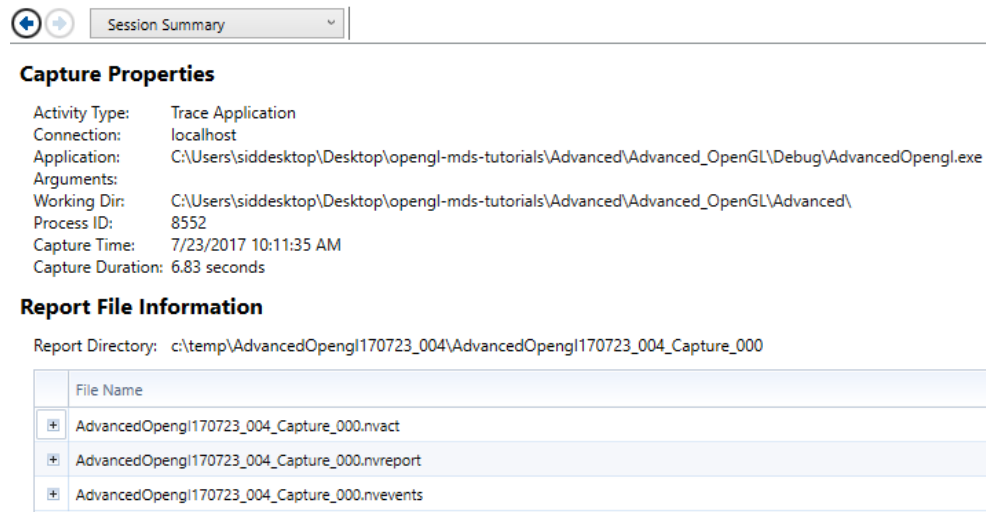
GPU 0 GeForce GTX 970

Utilization (%)	Min	Avg	Max
0.0			

⚠ Device context 0xC6011521 on render context 0x20000 has 127 CPU frames, but only 124 GPU frames.

Profiling

- Click the session summary, timeline to get a more detailed report.
- Session Summary – Capture time, duration and report directory



The screenshot shows a software interface for session summary. At the top, there is a navigation bar with a left arrow, a right arrow, and a dropdown menu labeled "Session Summary". Below this, the "Capture Properties" section lists various details about the captured session. The "Report File Information" section shows the report directory and a table of files generated during the capture.

Session Summary

Capture Properties

Activity Type: Trace Application
Connection: localhost
Application: C:\Users\siddesktop\Desktop\opengl-mds-tutorials\Advanced\Advanced_OpenGL\Debug\AdvancedOpengl.exe
Arguments:
Working Dir: C:\Users\siddesktop\Desktop\opengl-mds-tutorials\Advanced\Advanced_OpenGL\Advanced\
Process ID: 8552
Capture Time: 7/23/2017 10:11:35 AM
Capture Duration: 6.83 seconds

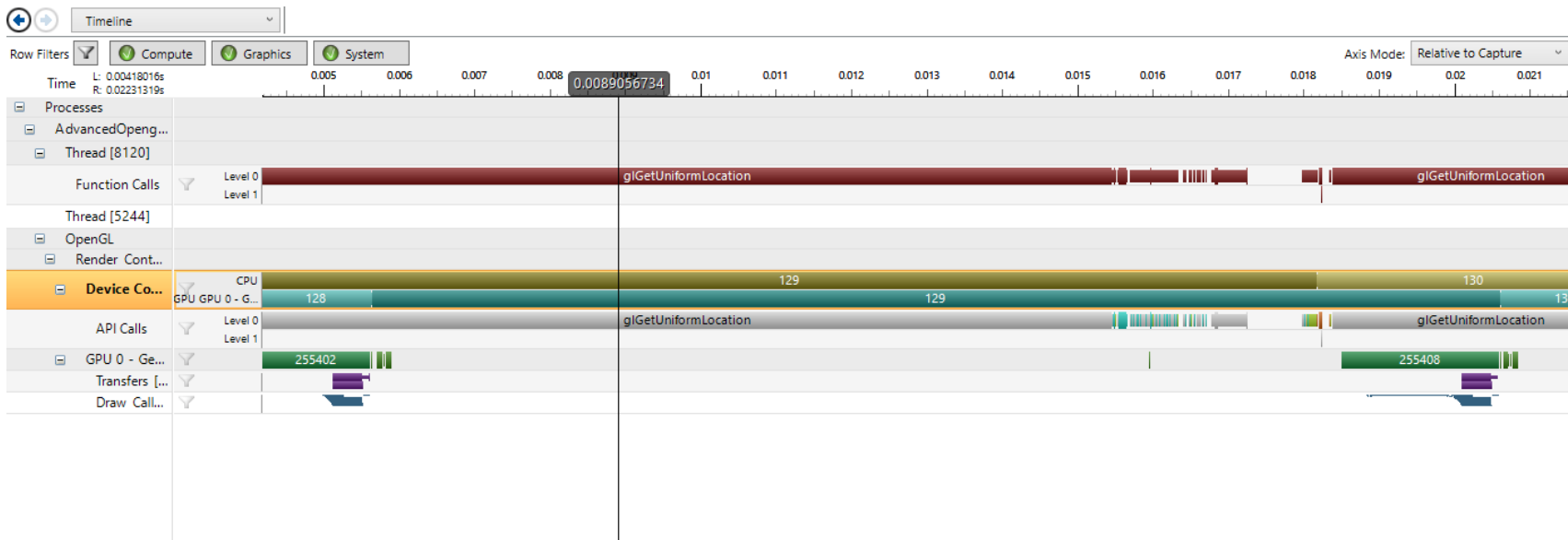
Report File Information

Report Directory: c:\temp\AdvancedOpengl170723_004\AdvancedOpengl170723_004_Capture_000

File Name
AdvancedOpengl170723_004_Capture_000.nvact
AdvancedOpengl170723_004_Capture_000.nvreport
AdvancedOpengl170723_004_Capture_000.nvevents

Profiling

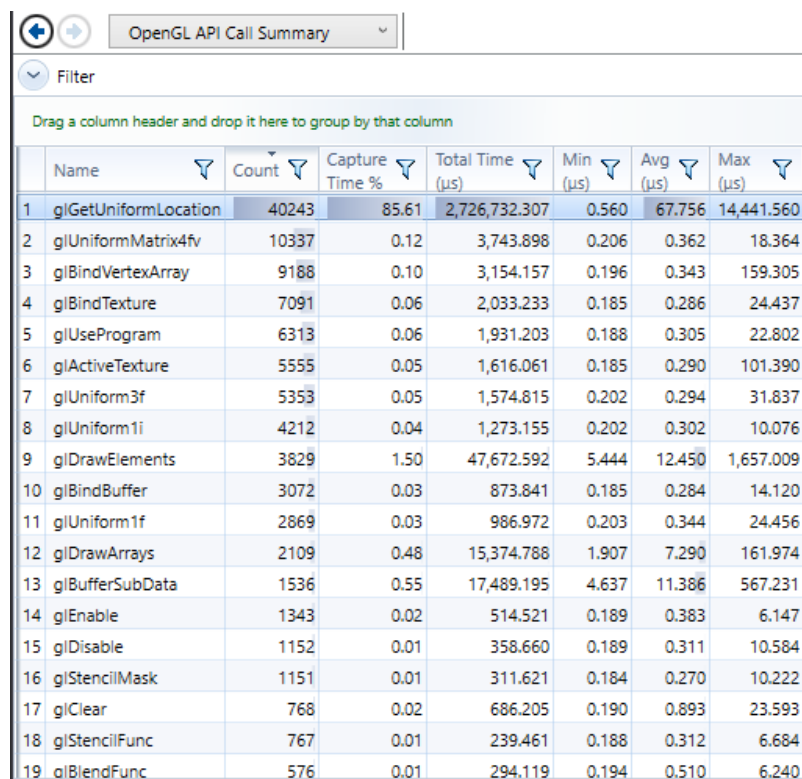
- Timeline – CPU and GPU Usage for the duration of the capture



- Observe
 - 129 is the frame number being processed
 - GPU lags behind CPU
 - Time taken by glGetUniformLocation compared to other API calls

Profiling

- API call summary – Number of times an API call was made and total time taken.



The screenshot shows a software interface for profiling OpenGL API calls. At the top, there is a title bar with a back arrow, a forward arrow, and a dropdown menu labeled 'OpenGL API Call Summary'. Below this is a 'Filter' section with a dropdown arrow and a green instruction text: 'Drag a column header and drop it here to group by that column'. The main part of the interface is a table with 19 rows of API calls. Each row has a number in the first column, followed by the API call name, and then five columns of performance metrics: Count, Capture Time %, Total Time (μs), Min (μs), Avg (μs), and Max (μs). The table is sorted by Total Time in descending order, with 'glGetUniformLocation' being the most frequent and longest-running call.

	Name	Count	Capture Time %	Total Time (μs)	Min (μs)	Avg (μs)	Max (μs)
1	glGetUniformLocation	40243	85.61	2,726,732.307	0.560	67.756	14,441.560
2	glUniformMatrix4fv	10337	0.12	3,743.898	0.206	0.362	18.364
3	glBindVertexArray	9188	0.10	3,154.157	0.196	0.343	159.305
4	glBindTexture	7091	0.06	2,033.233	0.185	0.286	24.437
5	glUseProgram	6313	0.06	1,931.203	0.188	0.305	22.802
6	glActiveTexture	5555	0.05	1,616.061	0.185	0.290	101.390
7	glUniform3f	5353	0.05	1,574.815	0.202	0.294	31.837
8	glUniform1i	4212	0.04	1,273.155	0.202	0.302	10.076
9	glDrawElements	3829	1.50	47,672.592	5.444	12.450	1,657.009
10	glBindBuffer	3072	0.03	873.841	0.185	0.284	14.120
11	glUniform1f	2869	0.03	986.972	0.203	0.344	24.456
12	glDrawArrays	2109	0.48	15,374.788	1.907	7.290	161.974
13	glBufferSubData	1536	0.55	17,489.195	4.637	11.386	567.231
14	glEnable	1343	0.02	514.521	0.189	0.383	6.147
15	glDisable	1152	0.01	358.660	0.189	0.311	10.584
16	glStencilMask	1151	0.01	311.621	0.184	0.270	10.222
17	glClear	768	0.02	686.205	0.190	0.893	23.593
18	glStencilFunc	767	0.01	239.461	0.188	0.312	6.684
19	glBlendFunc	576	0.01	294.119	0.194	0.510	6.240

Profiling

- Open Gl Api calls – Breaks the calls on per process id.

Filter

Drag a column header and drop it here to group by that column

	Name	Start Time (μs)	End Time (μs)	Duration (μs)	Render Context ID	Device Context ID	Draw Call ID	Transfer ID	Dispatch ID	Process ID	Thread ID
1	glGetUniformLocation	1,668,044.281	1,682,485.841	14,441.560	0x20000	0xC6011521				8552	8120
2	glGetUniformLocation	167,842.026	182,032.527	14,190.501	0x20000	0xC6011521				8552	8120
3	glGetUniformLocation	767,827.571	782,011.326	14,183.755	0x20000	0xC6011521				8552	8120

- OpenGL Draw calls – Draw Call ID per process ID.

	Draw Call ID	API Call	API Start Time (μs)	API Duration (μs)	Latency (μs)	GPU Start Time (μs)	GPU Duration (μs)	Command Buffer ID	Render Context ID	Device Context ID	Process ID
1	4675	glDrawElements	366,266.356	1,657.009	4,917.596	371,183.952	46.852	255535	0x20000	0xC6011521	8552
2	5818	glDrawElements	982,798.070	179.391	4,122.791	986,920.861	7.937	255758	0x20000	0xC6011521	8552
3	5084	glDrawElements	583,306.302	176.374	3,971.087	587,277.389	407.868	255613	0x20000	0xC6011521	8552
4	8314	glDrawArrays	2,320,482.605	161.974	3,231.182	2,323,713.787	5.953	0	0x20000	0xC6011521	8552
5	4416	glDrawArrays	232,136.887	121.111	3,179.828	235,316.715	15.490	255486	0x20000	0xC6011521	8552

Profiling

- OpenGL Frames – Gives framewise summary
- OpenGL Transfers – Provides duration for each of the transfers
- Function Calls – Gives details of CPU function calls
- GPU Devices – Provides information regarding the gpus used.

Resources

- **Check out the videos from** <https://developer.nvidia.com/nsight-visual-studio-edition-videos>
- **Watch**
 - Profiling OpenGL 4.2 with NVIDIA Nsight Visual Studio Edition 3.2
 - Debugging OpenGL 4.2 with NVIDIA Nsight Visual Studio Edition 3.2
- **User Guide**
 - http://docs.nvidia.com/nsight-visual-studio-edition/Nsight_Visual_Studio_Edition_User_Guide.htm#Nsight_Visual_Studio_Edition_User_Guide.htm%3FTocPath%3D1
- **NVTX Library (Extension tools library for markers)**
 - http://developer.download.nvidia.com/NsightVisualStudio/2.2/Documentation/UserGuide/HTML/Content/NVTX_Library.htm

Exercises

- Run Nsight on your application
- Debug the application
- Generate trace capture report.